SpaceFusion: A Multi-Server Architecture For Shared Virtual Environments

Hiroyasu Sugano* Koji Otani Haruyasu Ueda Shinichi Hiraiwa Susumu Endo Youji Kohda

Fujitsu Laboratories Ltd.

Abstract

We propose a scalable architecture for shared virtual environments, named *SpaceFusion*, which is designed to provide practical, large-scale services on the Internet. Here, the world is divided into smaller chunks, called regions, and a region plays a central role to obtain scalability and to realize the notion of Fusion. *SpaceFusion* clients can get information on several regions from possibly different servers at once, and the information is fused and presented on the browser. We also introduce *Chiba*, the prototype system currently being developed using Java and the Liquid Reality VRML library for Java.

CR Categories. H.5.1 Multimedia Information Systems; H.5.3 Group and Organization Interfaces; I.3.7 Three-Dimensional Graphics and Realism: *Virtual Reality*

Keywords. Distributed Shared Virtual Environment, Client/Server Model, Scalability, VRML

1 Introduction

The recent pervasive explosion of the World Wide Web has shown the possibility of the Internet as an infrastructure for building a global interactive virtual environment, or Cyberspace. Exploring the possibility, an open process to standardize the Virtual Reality Modeling Language (VRML) has started and is being intensively developed on the Internet. Steady progress has also been made in research on Distributed Virtual Environments, and some have been implemented as experimental services on the Internet.

While the notion of a cyberspace has been described in a variety of ways, the WWW explosion made this a little more concrete. We believe that the cyberspace of the future will not be just a virtual world, but it will be combined together with the real world synergistically. More specifically, we claim that cyberspace will be an aggregate of digital cities, some of those mirroring real cities in the following sense [7]:

- Cyberspace will mirror the external appearance of real cities.
- Cyberspace will mirror the social activities in real cities as well. It will serve as a 3D information viewer of real cities and it will also support communities in real/virtual life.
- Cyberspace will be customized for individuals to support their everyday behavior. It will help each individual based on his/her personal tastes or preferences.

The scope of this paper mainly lies within the second aspect of cyberspace. Here, people can obtain useful information, make reservations for concert seats, or contact their friends.

In this paper, we propose a scalable architecture for a shared virtual environment, named *SpaceFusion*, which is designed to provide practical, large-scale services in virtual space. Such services should be integrated with various information sources in real cities, those within virtual space, and some multimedia communication services taking advantage of virtual space. To realize this, our *SpaceFusion* architecture allows simultaneous access to multiple servers from clients. Users can select relevant information sources, and the provided set of information is fused and presented to them.

Technically, *SpaceFusion* is an object-oriented architecture based on a client/server model which allows multiple server access. To solve the server bottleneck, it adopts region-based communication filtering and client-side communication control with the notion of Aura. The notion of regions in *SpaceFusion* are similar to that of *locales* adopted by Spline[2], but has differences in details. The 3D objects in *SpaceFusion* are called entities and can be shared by multiple users with a mechanism of proxies. Entities can have their own behaviors, some can be moved, and their ownership can be transferred. Avatars are also special entities. *SpaceFusion* is currently being refined within our research project, and we are developing a prototype called *Chiba* to show and evaluate the feasibility of our architecture.

In our paper, we present the basic concepts of the *SpaceFusion* architecture and scenarios which will illustrate our motives in Section 2. Section 3 is an overview of the architecture and protocols in *SpaceFusion*. In Section 4, we briefly introduce the prototype *Chiba* we are currently developing. We will discuss the relationship with other works in Section 5 and conclude in Section 6.

2 Basic Concepts and Scenarios

2.1 Issues To Be Addressed

When we provide practical, large-scale services in virtual space, we have to address three important issues: performance, management, and stability.

Performance: This issue arises due to the capacity of the machines, the network bandwidth, and the type of communication and protocols. Assuming a client/server architecture with a single server, if this server provides all the communication services,

* 1-9-3 Nakase, Mihama-ku, Chiba-shi, Chiba 261, Japan
suga@iias.flab.fujitsu.co.jp

this causes significant scalability problems on the number of participants. Moreover, as 3D virtual space models are usually huge, if it serves an entire 3D world model, it can no longer be scalable. In contrast, a completely distributed architecture without servers can scale more, but the bandwidth of the network to which the client computers are connected is critical and the communication protocol must be carefully designed to reduce the network traffic. We have to balance the overall load to reduce the network traffic and the computation load on the servers.

Management: This is a more serious matter in providing practical services because it is related to the flexibility and extensibility of the whole system. Practical services will be composed of various information sources, and in general they are possessed by different organizations. For example, a local newspaper company knows about local affairs in the city, while a weather forecasting company knows of the weather forecast for several cities.

The entire city information cannot be covered by one information source or one sole organization. Thus it is unrealistic to put it into a single server from the viewpoints of both server administrators and information providers. For the administrators, registering and maintaining the practical information of all the services is too much of a burden. For the information providers, it is much better to hold the information at their sites than to give it to the server for the security reason.

We also expect that an architecture providing such services be open, that is, new application services can be easily added and the present services can be modified or withdrawn without much difficulty.

Stability: This is related to the stable usability of the system for both users and administrators. One of the important problems on stability is fault-tolerance because, for practical services in particular, partial errors or deficiencies must not cause the whole system collapse. Another important issue in practical services is security. Virtual space on the Internet is a public area and the system architecture should ensure security at an appropriate level.

2.2 Basic Concepts

To solve the issues described above, we designed the *SpaceFusion* architecture as follows:

- A client/server model,
- A number of information services provided by different organizations at their own servers,
- Selective integration, or *Fusion*, of these information sources at the user side based on his/her interests.

2.2.1 Client/server model

As stated above, a completely distributed architecture without servers is more scalable than a server-centered architecture, but it burdens each host with unnecessary communication overheads. This also raises the problem of security because every host must provide a direct connection to another host if it wants to communicate with it.

By adopting a client/server architecture, a mechanism for filtering unnecessary communications can be incorporated at the servers. This can also ensure an appropriate level of security because users can restrict personal information to be open to the servers, and do not allow direct connections from other hosts which they do not want.

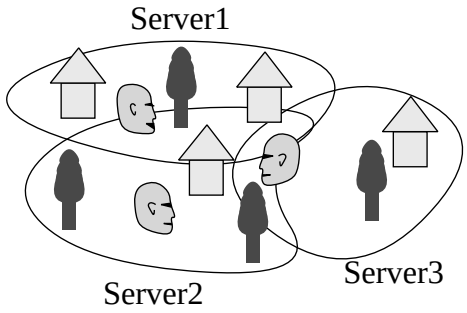

Figure 1: Fusion of various information from different servers.

2.2.2 Various information services by different servers

The management issue above tells us that a single server approach will not suit practical, large-scale services. Therefore, it is quite reasonable to assume that different servers will provide their own information services on the Internet. Our architecture is designed based on this assumption, and users will simultaneously select several servers to get the information they want.

This also suits the decentralized nature of the Internet. The success of the WWW is attributed to the fact that anyone can become an information provider, that is, there is an equal opportunity for producers and consumers. Our *SpaceFusion* architecture aims for a similar degree of availability. As a WWW server, a *SpaceFusion* server can be easily set up by information service providers themselves. Because this does not assume any centralized managers, this will not cause serious management problems, and more importantly, this conforms to the Internet policy.

SpaceFusion clients can connect to several servers simultaneously, and information from different servers are amalgamated and presented on the browser. This is the notion of Fusion.

2.2.3 Fusion of multiple information sources

A concept of fusion in *SpaceFusion* architecture has three aspects.

Horizontal fusion: This is the fusion of small spaces provided by possibly different servers, resulting in the seamless construction of a large space.

Vertical fusion: This is the fusion of spaces provided by possibly different servers, resulting in the overlaying of various information on the same space.

Hyper fusion: This is the fusion of virtual space and information from the real world.

Horizontal fusion allows spatial partitioning into smaller chunks, each of which is managed by possibly different servers. Thus, it enables local organizations to hold information about the local areas by themselves. By vertical fusion, we can distribute different kind of information into the appropriate servers owned by different organizations. Hyper fusion is another kind of fusion, and this fuses the virtual world and the real world. This must be a key concept of the cyberspace in the future.

By these fusion, geographical information, traffic information, and weather information provided by different organizations can be viewed in the same scene. Figure 1 depicts a world where three servers provide each parts of data. Consistency of the world data does not have to be assured in the architecture level. This is an application-level issue, and it is the responsibility of the clients to select appropriate servers.

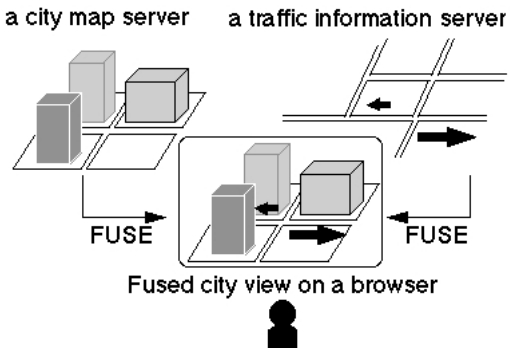

Figure 2: Fusing several regions from different servers.

2.3 Scenarios

The following two scenarios will best describe the power of *SpaceFusion*.

I. Walking in Digital Town TOGETHER

1. Fujiko and Michiko are young ladies living in Chiba city and they are good friends on the network. They often meet in Digital Chiba and enjoy the digital life together, but they have never met in real even though both live in the same city. Today, Michiko invites Fujiko to her home in Digital Chiba. As usual, they enjoy chatting.

2. Michiko says, "By the way, I have just accepted a marriage proposal. Please come to my wedding party if you can. I will introduce my fiance and *myself* to you." Fujiko quickly replies, "Congratulations! I'll definitely go. It's a good chance to meet each other in person." Michiko sends the party invitation. Fujiko receives and merges the party schedule into her personal schedule.

3. "But I have never been to the restaurant," Fujiko continues, looking at her schedule board on the screen. "I may get lost. How can I reach the restaurant from Chiba central station?" Michiko replies immediately, "OK, I'll guide you to the place now."

4. Michiko fetches several recommended routes by contacting a city guide server which offers customized traffic information to the contractors. There is a charge for the service, but Michiko often uses it because of its good quality. The recommended routes illustrate several ways from Chiba central station to the party place in Digital Chiba. Michiko selects a route and begins to walk along it with Fujiko.

In this example, two city servers[1] are used. One city server maintains a collection of regions which forms the skeleton of Digital Chiba. The other city server reports traffic information as a big region. The big region is FUSED over the regions of Digital Chiba by *SpaceFusion* as in Figure 2. Region access information could be transferred from a browser to another and thus the traffic information could be shared between people.

[1] A server in *SpaceFusion* is sometimes called a *city server* because cities are suitable units provided by servers.

II. Shopping for Household Goods AT HOME

1. Michiko and her fiance are now preparing for their new life. First of all, they rent an apartment from a real estate agent. They also get a precise 3D model of the room from the agent. They now plan to buy a nice carpet for their new living room. As they are Internet-age persons, they immediately enter Digital Chiba to shop.

(Intermission) If you have a favorite shop, then you can go to the shop directly. Otherwise, you must find an appropriate shop, but it is a waste of time to tour virtual shops in a digital city. Open Community proposal[1] describes a mechanism for selecting a shop in a virtual mall. In the proposal, a virtual mall could have an auction mechanism whereby the right to open shops in front of the customers are sold to potential virtual shop owners. When the customers stand at the front gate of the virtual mall, they will see the shop of the auction winner. However, there is no assurance for the customers that this shop is the best for their shopping needs. The problem is that the process of the auction is not visible to them and there is no chance for them to take part in the process. We propose a new way for shopping in the digital space that alleviates this problem.

2. First of all, Michiko and her fiance set up the 3D model of the new room in Digital Chiba as a shopping spot. As they want a carpet to fit the room, it is the best place for shopping in this case. There is no need to measure the room's length and width.

3. Then they contact a yellow pages server and get a list of carpet shops. They select two or three shops and call up the shops' *salesavatars* to the room, more precisely, to its 3D model where the avatars of Michiko and her fiance are. The negotiation takes place between those avatars.

4. The salesavatars greet them and notice the existence of competitors. To win business they must have more aggressive and flexible bargaining strategies than their competitors. Michiko tells them, "We want to buy a nice carpet to make this room feel comfortable."

5. The salesavatars create customized sales floors to display their carpets to recommend, and the newly created sales floors are fused with the room where the avatars are standing. Michiko, her fiance, and the owner salesavatar can see the fused sales floors. But other competing salesavatars cannot see their rivals' sales floors, though they can see the negotiation process or on-going conversation between the customers and the competitors. This configuration can keep the adopted bargaining strategies secret from the competitors.

6. Michiko and her fiance evaluate the recommended carpets one by one in their room, checking the fitness of each carpet to the room with their own eyes. Note that they can directly compare carpets in one shop with those in another. This benefit is never achieved in real shopping, or even by the Open Community approach. Our approach is quite similar to catalog shopping, but is more sophisticated.

7. Finally they decide to buy one of the recommended carpets. This means they also get its 3D model. The carpet itself will come soon, but even before the arrival, they can spread the carpet in the room trying the best layout.

In this example, the regions of a room and the sales floors in the shops are FUSED by *SpaceFusion*, while salesavatars

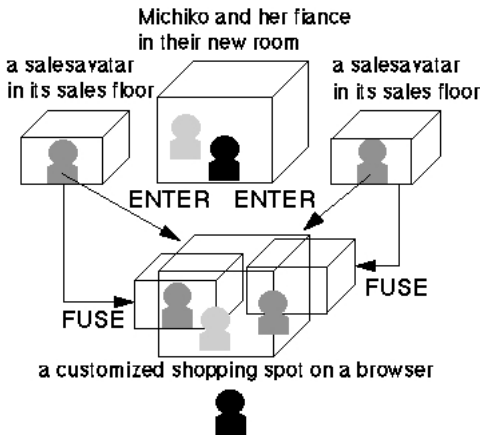

Figure 3: Creating a customized shopping spot.

just ENTER or visit the room region, as illustrated in Figure 3. Therefore the salesavatars can see and talk to one another and each salesavatar can see its own sales floors, but cannot see those of rivals.

3 Overview of *SpaceFusion* Architecture

Now we describe the technical details of the *SpaceFusion* architecture.

3.1 Entities and Regions

In *SpaceFusion*, an object is the unit of computation and communication. Thus it is an object-oriented architecture and it actually implements management of a distibuted object-oriented database as in Spline [2].

Entities. Among all the objects which constitute the architecture, the objects appearing in the world are called *entities*. All visible objects, including buildings, pieces of furniture, cars, and avatars are entities. Invisible objects without 3D shapes can be entities as well. Each entity is owned and controlled by a client.

Entities can send and receive messages, they can have their own behaviors, and they can migrate. Thus they are units of communication, behavior, and distribution in *SpaceFusion*. An entity is composed of a VRML 3D model, its behavior described in a scripting language like Java, and other attributes including its name, creator, avatar/product information, and so forth. The state of an entity includes the region containing it, its position and orientation in the region, and its owner.

Regions. Even though we adopt a multiple server architecture, each server will easily cause a bottleneck if the entire world served by it is dealt with as one chunk. To solve this problem, we partition the world into several smaller chunks called *regions*.

Regions are linked to each other with a contiguity relationship, which gives the region its neighbors, possibly not spatially contiguous (Figure 4). The relation is usually defined statically, but some regions An entity in the world is contained in exactly one region, and before that, it must *enter* the region. As some kind of entities like avatars and bots can move around, such an entity may dynamically change the region containing it. Thus, it may enter a new region and exit from the old one. Only entities

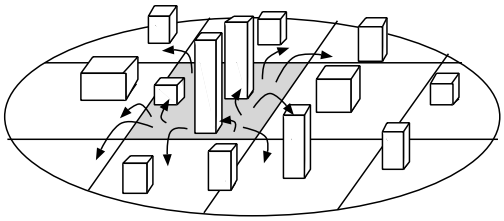

Figure 4: Contiguity relationship among regions.

contained in a region can be displayed in a client as contents of the region.

A region plays an important role in *SpaceFusion*, because it serves a primary unit of fusion. A region is a unit of communication and sharing of the world, thus the world data is sent to each client per region and the amount of the data sent at once is greatly reduced. The position and orientation information of avatars or bots is exchanged only among clients watching the region in which they are contained. This reduces the communication traffic between the server and the clients very much.

The notion of regions has been used by several researches on virtual environments. The notion of *locales* in Spline [2], or regions in Open Community proposal [1], is very similar to our notion of regions. It also serves a unit of communication, and each locale corresponds to a multicast address for communication among the locale and the objects in it. We do not adopt IP multicast because it is not reliable, and it may cause a serious consistency problem. Spline gives the visibility a primary importance in defining a neighboring relation. It is useful to give the relation automatically, and we can use such a mechanism in the world creation phase. NPSNET[9] also uses similar notion of regions, called Area Of Interest, but our notion is more flexible. *SpaceFusion* allows different regions to occupy the same space, in this case, a client just select either to watch. A region can be attached to an entity to realize a nested region which can move like tour buses.

3.2 Clients and Servers

It is useful to divide the function of a client to clarify it. A client plays two main roles in our architecutre. One is to receive the data of the world, or to just *watch* the world, and the other is to let its entities participate into the world, or in other words, to *deliver* entities. Remember that an entity is owned by one client.

A *watcher* is a client which monitors the worlds. It is a major role of browsers. A *deliverer* is a client as an owner of entities and it controls these entities in the worlds. A browser which can be used to control an avatar has this aspect. Usually a client or a browser in *SpaceFusion* has both capabilities, but a client with just one capability is also allowable. A client that acts just as a watcher can obtain information from a world, like ghost-mode in Habitat. A client that acts just as a deliverer provides world data and services. The latter may be a typical client of an information service provider.

Check-in and check-out: watching regions. In order to obtain the world information, or to watch the world, a client must check in to regions. Here *check-in* and *check-out* are primary notions in protocols between clients and regions. When a client takes part in the world, it checks in to the region of the entry point first, and then it obtain the world data.

The essence of the notion of Fusion lies in just a simultaneous

watch to multiple regions possibly provided by different servers. By checking in the regions contiguous to the current one, a client can watch several regions at the same time as a single large space. Users watching the scene will not usually realize the region borders. As the viewpoint/avatar moves, it may go across a region border. Then, the client may check in a new region and check out another region it has been watching. The client repeatedly checks in and checks out according to the movement of the viewpoint/avatar, and a user just experiences continuous changes in the world scene.

Connecting to servers. When a client connects to a server at first, after obtaining the host name and the port number of the server, it tries to establish a TCP connection with the server. Having established the connection, they exchange identification information. Connections to different servers are allowed, and the disconnection from some of those is possible at any time.

Because we want entities to be able to move among several worlds served by different servers, they must be globally identified. To do that, our architecture gives a *real name* to an entity, which is globally unique. The real name is given to an entity when its owner client connects to a server for the first time using the identification information. Given a real name, an entity can exist in the *SpaceFusion* worlds.

A server and a client establish a connection for *SpaceFusion* protocol, which will be explained later, when the corresponding *client proxy* object is created in the server and the corresponding *server proxy* object is created in the client. The server and the client communicate with each other through these proxy objects.

Proxies: sharing entities. As stated above, each entity is owned by a client in *SpaceFusion*. The owner client of an entity has its *substance*, the object with all the information of the entity. Before the client connects to the server, this entity exists only in the client as the substance object. Entities can be shared by several clients by distributing their *proxy* objects to the servers and the clients. When a client owning an entity connects to a server, the entity creates its *master proxy* on the server and puts it in a required region server. Clients watching the entity must have its *slave proxy*.

Figure 5 illustrates the server and clients connect through the client/server proxy objects and how entities are shared using master/slave proxies.

Aura management by clients. In addition to adopting regions, another point to resolve the server bottleneck is Aura management at the client side. The traditional client/server design of a distributed virtual environment takes complete control of communications at the server side, causing a server bottleneck. Since we have seen a drastic increase in PC computing power, we should transfer some of the duties to the client PCs.

The notion of Aura is used in DIVE [4] and Community Place [5, 8]. It controls communication among avatars and other objects based on their capabilities, their own states and the distances among them. Although it enables a reasonable communication control among entities, but its computational cost is usually expensive. It is reasonable to move this functionality to clients.

As a client watching a region knows the position and other Aura-related information of the entities within it, it can calculate the distances between these other entities and the client's avatar. Note that the distance is usually calculated in consideration of some external and internal conditions of avatars/entities. If the distance is sufficiently small, it sends more detailed information like gestures and facial expressions to the corresponding client.

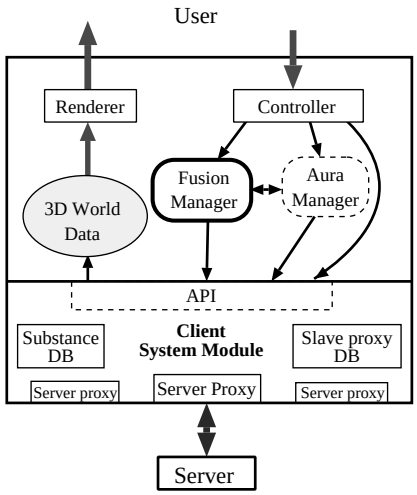

Figure 6: The Structure of a Client.

3.3 *SpaceFusion* Protocols

SpaceFusion has a set of object-oriented protocols to realize shared environments and various operations within these environments. The Basic Object-Oriented Protocol (BOOP), which is built on TCP/IP, provides a basic mechanism to send and receive messages between objects in servers and clients. The *SpaceFusion* protocol (SFP) is built on the BOOP. This protocol realizes the essential functions of the architecture, including clients watching regions, entities moving around regions, and sharing entities. Working with the basic components like substance/proxy objects of entities, region managers, and fusion managers, it embodies the 3D shared virtual environment.

We assume the *SpaceFusion* Application Specific Protocol built on BOOP and SFP. This protocol will be provided by application creators, and it can utilize the external protocols. The direct communication between clients is allowed at this level.

Structure of a client and a server. A client consists of a client system module, a fusion manager, world data and a renderer, a controller, and an Aura manager (Figure 6). The client system module is an object base manager, and manages the substances and slave proxy objects of entities. It also establishes connections to servers, and controls communication with servers using the server proxy objects. A fusion manager manages regions it is connecting to, i. e. watching, and the region where its avatar is. According to the motion of the viewpoint and the avatar, it decides when this client checks in to and out of regions, and whether or not to make the avatar enter the new region. A controller gives a mechanism of user interface to control the avatar and send messages to entities. An Aura manager checks the collision of Auras of the avatar and other entities based on the type of Aura.

A server consists of a server system module, a city manager, a set of region managers, and a name server (Figure 7). A server system module manages connections to clients, communication with clients using client proxy objects, and it controls master proxies of entities contained in the regions it has. A city manager manages the regions it has, and each region manager administers the entity data such as the positions and the entity's entry into and exit from the region. A name manager manages the correspondence of the names of entities and their ID's.

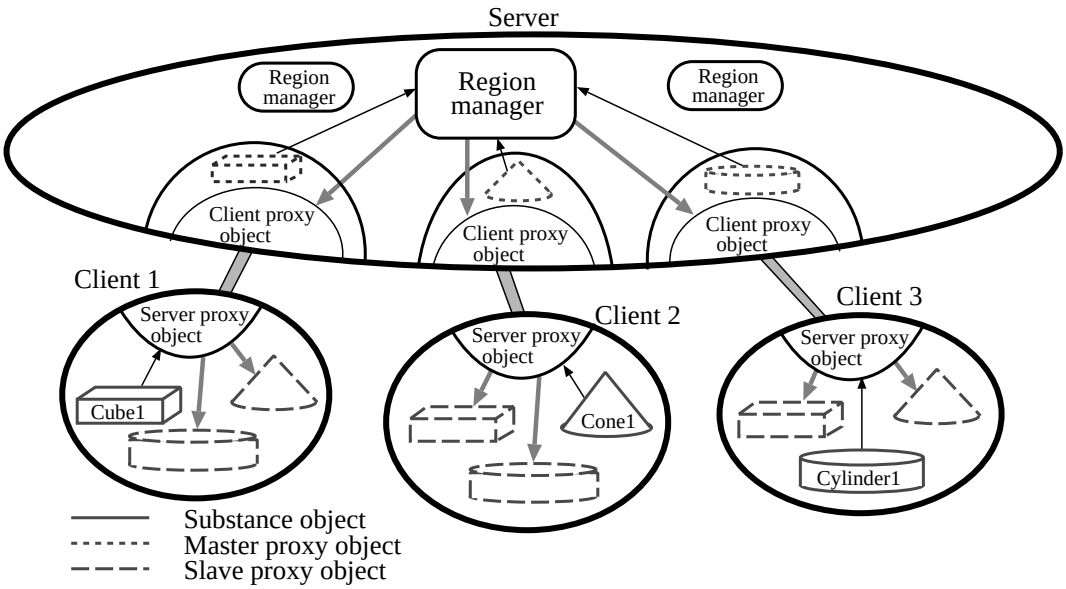

Figure 5: Sharing entities.

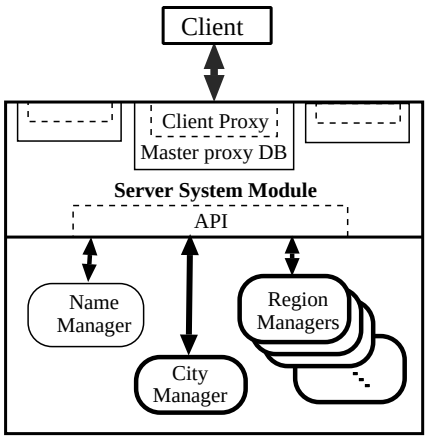

Figure 7: The Structure of a Server.

Check-in protocol. When the fusion manager of a client decides to check in to a region, it orders the system module to send a checkInRegion message to the serer with the ID of the region as its argument. When the server system module receives the message, it orders the region manager the check-in. Then, the region server sends messages to the client to create all the slave proxies of the entities contained in it, and the client can watch the region. When a client checks out a region, it asks the region server to check out, and the region server send messages to the clients to delete the slave proxies of entities in it.

When a client connects to a server, it establishes the connection as stated above, and it checks in some regions in a server as a watcher. It also behaves as a deliverer, thus sending messages to create master proxies of the entities it has. These messages must contain the identifiers for the VRML file of 3D models and/or program files for behavior scripts. Those files can be provided by URLs so that caches work effectively, or may be provided on CDROM.

Entering regions. When an entity enters a region, its master proxy object asks the new region for its entry. After granting permission to enter, the master proxy sends messages to all the clients watching the new region to create slave proxies of the entity. Having received the message, the clients create the slave proxies. The protocol for an entity to exit from a region is basically a similar procedure.

A client usually watches both regions which the avatar entity enters and exits from. In this case, an entity exits from the old region after entering the new region. Otherwise, the slave proxy of the entity in the client must be deleted once and it needs to be created again. The identity of an entity is always checked, and thus two slave proxies for the same entity never be created.

Communication between clients. When a client recognizes other avatars, it is watching the region in which these avatars are. If it has an avatar of itself, it can communicate with other clients owning these avatars.

A client watching the region has a substance of the avatar of itself and slave proxies of other avatars. It also gets information of the positions and orientations of avatars in the region from the region manager of the server. The Aura manager in the client calculates the distances of its avatar and other avatars, and decides which avatars can be accessible through the medium under consideration. To communicate with an avatar, a client sends a message to the slave proxy of the avatar, and it is forwarded to its substance in another client through the server. Figure 8 depicts how the clients communicate each other through the substances and slave proxies of entities.

4 *Chiba*: The Prototype System of *SpaceFusion*

We are currently developing a prototype based on the *SpaceFusion* architecture. The name is *Chiba*. *Chiba* is being developed using Java provided by Sun Microsystems and the Liquid Reality VRML library for Java released by Dimension X.

The development of *Chiba* is still continuing, so some of the functions described above have not been implemented yet. For example, the communication among entities, Aura management,

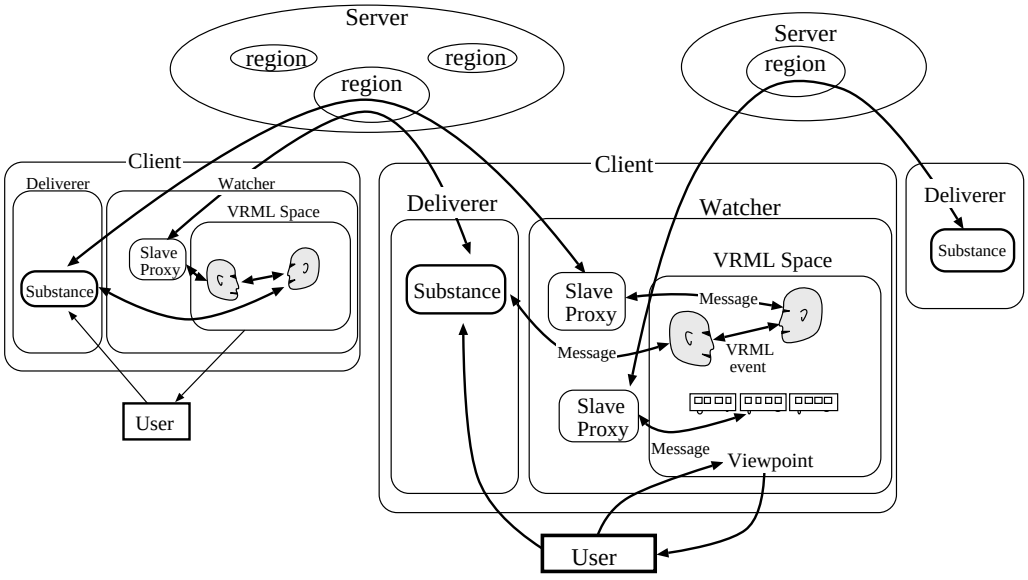

Figure 8: Communication between clients.

and regions attached to entities have not been implemented yet. The current prototype consists of the following components:

Server: This prototype only serves a fixed number of regions and the contiguity relationship among region are restricted to a spatially contiguous one.

Browser: The position and the orientation of the avatar, or the viewpoint, can be changed using the keyboard. A headlight function, a wide-view mode, selectable levels of rendering/texture quality, and connection to multiple servers have been implemented.

Server-embedded client: Clients embedded in the server process and provide scalable entities. As each entity must belong to a client, entities provided by a server like buildings and geographic models are served by a server-embedded client.

Entity: The following type of entities are currently implemented.

1. Basic entities, which are generic for all entities. The position and the orientation are can be shared among clients.
2. Scalable entities, whose size is also sharable. Immovable entities like buildings are usually this kind.
3. Bounce entities, which move around the world.
4. Human entities, which have human-like figures. An avatar is usually an entity of this type. At present, we have only robot-like avatars.

Figure 9 and 10 are screen images of *Chiba*. Each illustrates the example scenarios described earlier. Browers used in these examples are displayed in wide-view mode with three windows. The viewpoint orientations of the right and left windows make an angle of 45.0 degrees with that of the center window. The advantage of the wide-view mode browser is well understood by considering a group walk talking each other. We have implemented the key functionality of a server and a client, such as the fusion of regions provided by different servers. But, communication between clients is not implemented yet, thus the whole scenarios cannot be realized by the present system.

Figure 9 is a screen shot of Michiko's browser in the scenario "Walking in Digital Town TOGETHER". The main body of Michiko 's avatar is unseen, but her raised right arm is visible in the rightmost screen, which points in the direction to go. The leftmost avatar which looks in the direction is Fujiko's avatar. On the road, the selected recommended route is depicted as an overlay.

Figure 10 is a screen shot of Michiko's browser in the scenario "Shopping for Household Goods AT HOME", which shows a temporarily created shopping spot. An avatar in the center screen is her fiance and now checks a carpet spread in the room. The leftmost and rightmost avatars are salesavatars from two different shops. Their customized sales floors are placed next to the room.

5 Related Works

Our research is related to some of the important works on distributed virtual environments currently published.

5.1 Spline

Spline is a platform for a scalable shared virtual environment developed by MERL[2]. As stated earlier, it is a distributed object-oriented architecture and the notion of *locales* is very similar to that of regions in *SpaceFusion*. An experimental virtual world called Diamond Park is created based on Spline, which has notable features like the whole body interface with bicycles, rich audio interface including speech communication, and the obelisks which embody non-Euclidean connectivity between locales.

A locale is managed by a locale server, each of which has its own Multicast address for communication among the objects in the locale. Our *SpaceFusion* protocols are built on the TCP connection, which ensures the packet reachiability. Our choice costs a little heavier traffic, but consistency among the states of entities can be ensured, which is quite important in the practical services including electronic commerce. At the same time, however, we have to consider more seriously on the QoS of multimedia services.

The neighbor relationship of locales is statically defined mainly based on visibility. The contiguity relationship of regions

in *SpaceFusion* is usually defined statically as well, but, in some cases including regions attached to the moving entity, it can be determined dynamically. Visibility is not the only principle for the contiguity relationship, so jumping to other unkown places is allowed in our architecture. The notion of fusing multiple regions from different servers is a novel feature of *SpaceFusion*.

5.2 NPSNET

NPSNET[9] is a global simulator for military purpose implemented with the DIS protocol. The world is statically divided into chunks, based on the notion of Area of Interest. These are fixed throughout the simulation. An object in a region is shared by active replication by all the client in the region. A region is implemented to have a multicast address as in Spline. When a client changes something in a region, the change is broadcasted to all the browsers viewing the region via its multicast address. Dead reckoning mechanism is used to prepare the loss of the multicast packets and to reduce the network traffic.

Because this is limited to simulation purposes, there is no way to set the permission to control objects owned by other browsers. As region prefetching is possible, the user can see contiguous regions if necessary. However, from the nature of this division, it is impossible to make a more complex regional structure, for example, two different regions which occupy the same space which *SpaceFusion* allows.

5.3 DIVE

DIVE[4] is another pioneering work on distributed virtual environment. It was developed by SICS as an experimental system for CSCW utilizing 3D virtual environment. In addition to using the notion of Aura, more minute control with the concepts of focus, nimbus and awareness is incorporated for multi-user interaction. A whiteboard, a conference table, and a podium are interesting components for multimedia communication.

DIVE is based on peer-to-peer communication with no servers, and for sharing objects, DIVE uses active replication and reliable multicast protocols. This may be because DIVE was designed as a system on a LAN, thus it will not have much scalability when we think of a wide area service.

DIVE uses the notion of *worlds*. Each world corresponds to a multicast address, and the worlds are completely disjoint. As worlds are larger units than regions in *SpaceFusion* and DIVE does not have a prefetching mechanism of worlds, it will not be effective to reduce the latency in visiting worlds.

5.4 Community Place

Community Place[5, 8], which used to be called Cyber Passage, is a system for network virtual reality based on VRML2.0.

Community Place has three different types of 3D objects: A) an avatar, B) a usual 3D object, and C) a 3D object with an application object (AO). An avatar is controlled by a client and an object in a type C is controlled by an application object. Basically, Objects with AO correspond to entities in *SpaceFusion*. The states of these objects can be fully shared, and they can appear and disappear in the world dynamically.

The architecture of Community Place has no notion of regions, and it uses Aura to obtain the scalability. Aura can provide a good control mechanism of communication between clients, but the collision detection of Aura is performed by the server, which may limit the scalabitliy on the number of clients. Community Place provides World Location Service as a name service of worlds. This may be used to get the scalabitliy, but *SpaceFusion* provides more a flexible and effective mechanism: fusion of regions from different servers.

6 Conclusion

We have described a scalable multi-server architecture for shared virtual environments *SpaceFusion*. In *SpaceFusion*, the space of the world is divided into smaller chunks called regions, and they play the important role as units of fusing space and resolving the communication bottoleneck. Entities are basic objects in the 3D world. They can have their own behavior and move around regions and servers. This mechanism is realized by the *SpaceFusion* architecture and protocols. We have shown the overview of those in this paper.

We are developing a prototype system called *Chiba* based on the *SpaceFusion* architecture. While the current status of our research is in the beginning stage, some of the important mechanism are not implemented yet, including communication among participants, the Aura management, and regions attached to entities. The evaluation is very important in this kind of architecture for practical services, but it is difficult without a suitable methodology. We have just started building such a methodology for an evaluation of this kind of systems.

SpaceFusion will serve as a test bed for practical multi-user shared virtual environments. We are also investigating possibility of our contribution to VRML3.0 for shared worlds, in a file format describing entities and worlds, application interfaces, or a level of protocols.

References

[1] D. Anderson, D. Greening, M. Ma, M. Marvit, and R. Waters, "Open Community Overview", `http://www.merl.com/opencom/`, November 5, 1996.

[2] J. W. Barrus, R. C. Waters, and D. B. Anderson, "Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments" *MERL Technical Report TR95-16a* 1996.

[3] D. Gelernter, "Mirror Worlds: Or the Day Software Puts the Universe in a Shoebox... How it will Happen and What it will Mean.", Oxford University Press, 1991.

[4] O. Hagsand, "Interactive MultiUser VEs in the DIVE System," *IEEE Multimedia Magazine, Vol 3, Number 1,* 1996.

[5] Y. Honda et al., "Extending WWW to support Multi-user Interactive 3D environment," *VRML '95*, 1995.

[6] W. A. Kellogg and J. M. Carroll, "Making Reality a Cyberspace", In *Cyberspace: First Steps*, M. Benedikt editor, MIT Press, chapter 15, 1991.

[7] Y. Kohda and M. Sonobe, "Cyberspace on the Web: Mirror Worlds of Real Cities", *FUJITSU Sci. Tech. J.*, vol. 32, no. 2, December 1996.

[8] R. Lea et al., "Technical Issues in the Design of a Scalable Shared Virtual World," *Sony Technical Report SCSL-TR-95-039,* 1995.

[9] M. R. Maccedonia et al., "NPSNET: A Network Software Architecture for Large-Scale Virtual Environments," *Presence Volume 3, Number 4,* 1994.

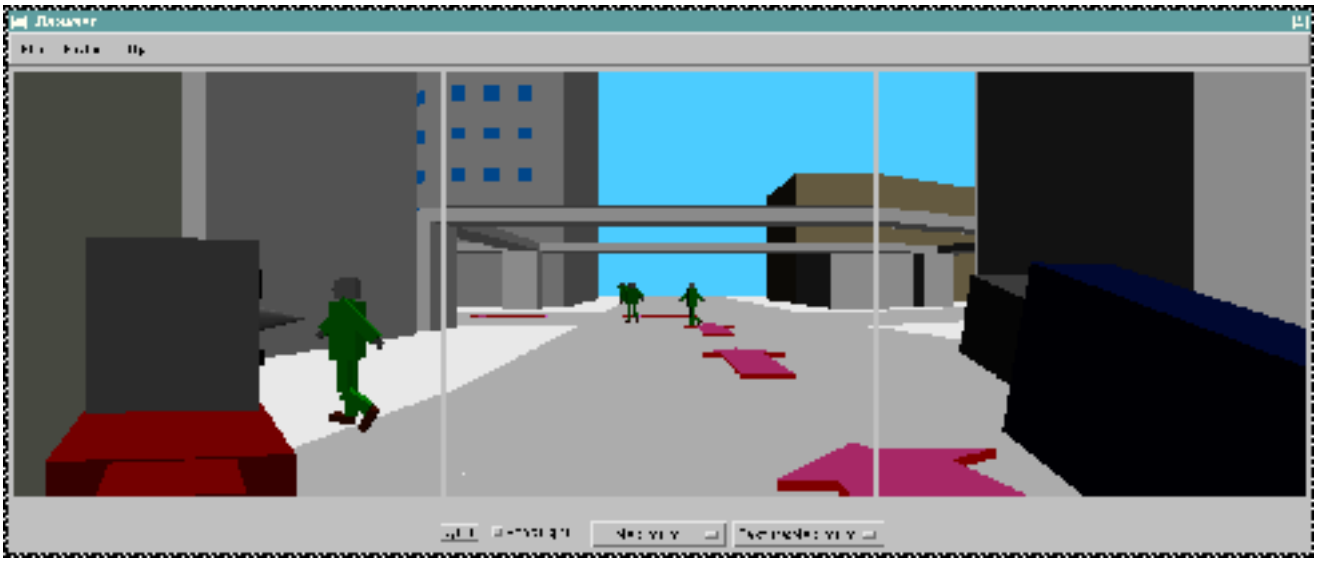

Figure 9: Screen image of *Chiba*: Walking in Digital Town.

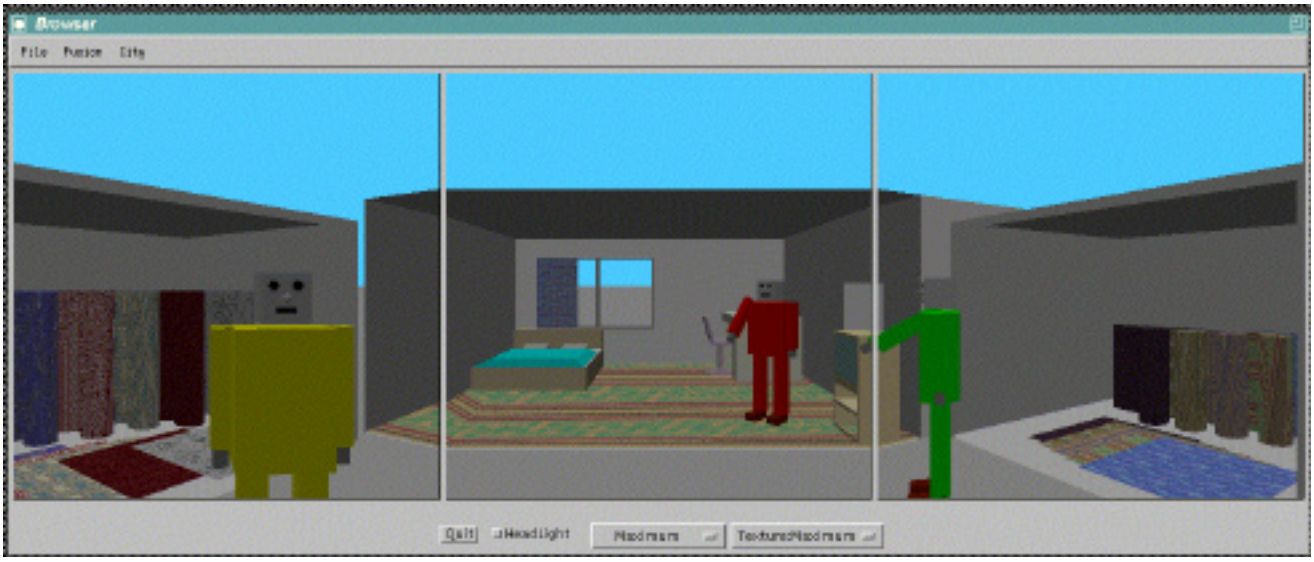

Figure 10: Screen image of *Chiba*: Shopping at Home.